PALADYN Journal of Behavioral Robotics

VERSITA

Review Article · DOI: 10.2478/pjbr-2013-0021 JBR · 4(3) · 2013 · 155-173

Stochastic Diffusion Search Review

Mohammad Majid al-Rifaie*, **John Mark Bishop**†

Goldsmiths College, University of London
New Cross Gate, London SE14 6NW, United Kingdom

Received 24-05-2013

Accepted 24-12-2013

Abstract

Stochastic Diffusion Search, first incepted in 1989, belongs to the extended family of swarm intelligence algorithms. In contrast to many nature-inspired algorithms, stochastic diffusion search has a strong mathematical framework describing its behaviour and convergence. In addition to concisely exploring the algorithm in the context of natural swarm intelligence systems, this paper reviews various developments of the algorithm, which have been shown to perform well in a variety of application domains including continuous optimisation, implementation on hardware and medical imaging. This algorithm has also being utilised to argue the potential computational creativity of swarm intelligence systems through the two phases of exploration and exploitation.

Keywords

swarm intelligence · resource allocation · optimisation · search · information exchange

1. Introduction

Noisy environments and/or incomplete data are often at the heart of hard real-world search and optimisation-related problems, generating input that established search heuristics (e.g. tabu search [45], simulated annealing [60], etc.) sometimes have difficulty dealing with [55]. Conversely, ever since their inception researchers have been attracted to the complex emergent behaviour, robustness and easy-to-understand architecture of nature-inspired swarm intelligence algorithms; and, particularly in challenging search environments, these algorithms have often proved more useful than conventional approaches [59].

This review paper surveys Stochastic Diffusion Search (SDS), a multi-agent global search and optimisation swarm intelligence algorithm based upon simple iterated interactions between agents. After considering SDS in the broader context of natural swarms, a high-level description of the algorithm is presented in the form of a search metaphor driven by social interactions. This is followed by an example of a trivial 'string search' application to illustrate the core algorithmic processes by which SDS operates, after which the development and detailed architecture of SDS are discussed in more detail. Then, in addition to analysing the behaviour of SDS and the possibility of embedding different agent-interaction strategies, the novel way in which SDS deals with computationally costly objective functions is investigated. Finally, various hybrid SDS algorithms are reviewed and issues related to various fielded applications of SDS presented. We conclude the review by highlighting key extant questions and current avenues of research.

*E-mail: m.majid@gold.ac.uk
†E-mail: m.bishop@gold.ac.uk

2. Swarm Intelligence

The story of the *blind men and the elephant* provides a simple illustration of how social interaction can lead to more intelligent behaviour. This famous tale, first set in verse by John Godfrey Saxe in the 19th century [91], characterises six blind men approaching an elephant. As their initial individual encounters with the elephant merely result in each experiencing only one aspect of the elephant's taxonomy, each person is led to entertain different ideas about the beast. E.g. One person met with the elephant's side and thought he had encountered a wall; another met with the elephant's tusk which he thought to be spear; a third found the trunk which he took to be a snake; the next met with the elephant's leg which he thought to be a tree; another found an ear which appeared to be a fan and the last tugged on the tail which he thought to be a rope. The moral of the story is to demonstrate how people build their belief framework based on partial evidence derived from incomplete knowledge about the world [59]. Conversely, if the blind men in the story had interacted with one another and communicated what they had individually perceived - wall, spear, snake, tree etc. - then each would rapidly have garnered enough evidence to conclude that they had actually encountered an elephant. Swarm intelligence formally demonstrates this move of realigning intelligence away from the individual towards the collective; its aim is to illustrate intelligent behaviour by considering individuals in a social context and monitoring their interaction with one another as well as their interaction with the environment [27]. Although some writers (e.g. [17, 44]) blur the difference between adaptation and intelligence by claiming that intelligence is actually the ability to adapt, writers in the field of swarm intelligence more characteristically emphasise that an individual is not merely an isolated information processing entity (e.g. [59]).

Natural examples of swarm intelligence systems that exhibit such forms of collective interactions are: fish schooling, bird flocking, bacterial growth, animal herding, nesting and foraging in the social insects etc. and in recent years, abstractions of such natural behaviour have suggested several new meta-heuristics for use in modelling collective in-

telligence. The simple and often successful deployment of these new meta-heuristics on traditionally difficult optimisation problems has in turn generated increasing interest in the nascent field of swarm intelligence algorithms: nature-inspired algorithms instantiating distributed computation via the interaction of simple agents and their environment (e.g. ant algorithms [38, 39] and particle swarm optimisation [57] etc.[1]).

In this paper we will illustrate Stochastic Diffusion Search - in which interactions between agents cause a population of agents to evolve towards potential solution states - and show that it shares many of the characteristics and behaviours of classical swarm intelligence algorithms; furthermore, the core stochastic diffusion process are illustrated in the behaviours of some social insects such as ants and bees (e.g. in locating food sources and nest site location). In the next two parts of the paper (Sections 2.1 and 2.2), we will explore SDS in this context.

2.1. Communication in Social Insects

Communication – social interaction or information exchange – as observed in social insects is important in all swarm intelligence algorithms, including SDS. Although as stated in [59], in real social interactions, not just the syntactical information is exchanged between the individuals but also semantic rules and beliefs about how to process this information; in typical swarm intelligence algorithms, only the syntactical exchange of information is considered.

In the study of interaction in social insects, two key elements are the individuals and the environment, which results in two modes of interaction: the first defines the way in which individuals interact with each other and the second defines the interaction of individuals with the environment [28]. Interaction between individual agents is typically carried out via agent recruitment processes and it has been demonstrated that various recruitment strategies are deployed by ants [52] and honey bees [47, 92]. These recruitment strategies may be used, for example, to attract other members of the population to gather around one or more desired areas in the search space, either for foraging purposes or in order to facilitate a colony relocation to a better nest site.

It has been observed that recruitment strategies in social insects may take the form of: localised or global recruitment; one-to-one or one-to-many recruitment; and may operate stochastically or deterministically. The nature of information exchange also varies in different environments and with different types of social insects. Sometimes the information exchange is quite complex and, for example, might communicate data about the direction, distance and suitability of the target; or sometimes the information sharing is relatively simple, for example, a stimulation forcing a particular triggered action. Nonehtless, what all recruitment and information exchange strategies have in common is an ability to distribute useful information across their community [72].

In the next section some different forms of information exchange are discussed in more detail and their relation to SDS recruitment strategies presented.

[1] N.B. As they are also 'nature inspired algorithms' described in terms of 'iterated populations of simple agents interacting with each other and their environment' the authors also partition evolutionary algorithms [18], genetic algorithms [46, 51], differential evolution [93] etc. as swarm intelligence algorithms, however this taxonomy is not currently reflected in the general swarm intelligence literature.

2.2. Methods of Communication

Chemical communication through pheromones forms the primary method of recruitment in many species of ants, however in one species, *Leptothorax acervorum*, a 'tandem calling' mechanism (one-to-one communication) is used. In this process the forager ant that finds the resource location, physically recruits a single ant upon its return to the nest, and by this action the location of the resource is physically publicised [75] to the population.

Conversely in group recruitment, one ant summons a group of ants, leading them to the resource location. Group recruitment may entail laying a pheromone trial from the resource to the nest; a more complex process in which the recruiting ant is no longer necessarily in physical contact with the recruited ants.

The most advanced ant recruitment mechanism is called 'mass recruitment' [32]; in this, worker ants both follow the pheromone trail and incrementally add an amount of pheromone on their journey towards the resource location. In such 'mass recruitment' the concentration of pheromone plays an important role in attracting other ants to the resource trail.

Different recruitment and communication algorithms thus induce different search performances. Ants communicating through group recruitment are faster than tandem calling ants, and similarly, ants utilising mass recruitment are more efficient in their performances than the former recruitment strategies [32]. Ant algorithms have been successfully applied to hard optimisation and search problems such as traveling salesman problem and the quadratic assignment problem [40].

However, as mentioned in [36], the success of the ants in reaching the food they have been recruited to obtain, varies from one species to another. In another form of communication, indirect or stigmergetic communication, the exchange of information is based on modifying the physical properties of the environment and its success lies in spatial and temporal attributes of mass recruitment and the positive feedback mechanism it employs. In this mode, which is based on using pheromone, short routes are loaded with more pheromone (because of the shorter time it takes the ants to travel back and forth between the source and target [42]).

An ant-like task allocation has been investigated in [62] where robots were used to simulate different non-communication and communication strategies, concluding that ant-inspired techniques of decentralised control, namely tandem-calling recruitment mechanism [75] shows better results than single robots doing the same task. This technique of information exchange is an instance of a broader type of recruitment strategy utilised in stochastic diffusion search [26], which will be discussed in more detail, later in this paper.

In honey bees the group recruitment is performed by means of waggle dances, in which the direction of the dance shows the location of the food source and the speed of the dance represents the distance to the target area. Each bee chooses one of the dancing bees as a guide to the food source.

In SDS, direct one-to-one communication (which is similar to tandem calling recruitment) is utilised. Although the recruitment behaviour of real ants is more complex than the behaviour in SDS, both are population-based and find their optima via agents communicating with each other. The effect of different recruitment strategies will be discussed later (see Section 5).

2.3. Search and Optimisation

In the swarm intelligence literature, search and optimisation are often used interchangeably. Nevertheless, search has been categorised into three broad types [74]:

- In the first definition, search refers to finding a (target) model in a search space, and the goal of the algorithm is to find a match, or the closest match to the target in the search space. This is defined as ***data search*** and is considered as a classical meaning of search in computer science [61].
- In the second type, the goal is finding a path (***path search***) and the list of the steps leading to a certain solution is what the search algorithm tries to achieve. In this type of search, paths do not exist explicitly but are rather created during the course of the search.
- In the third definition, ***solution search***, the goal is to find a solution among a large problem space of candidate solutions. Similar to the path search where paths do not exist explicitly, the search space consists of candidate solutions which are not stored explicitly but rather created and evaluated during the search process. However, on the contrary to the path search, the steps taken to find the solution are not the goal of the algorithm.

In optimisation, which is similar to the first search definition, the model is replaced with an objective/fitness function which is used to evaluate possible solutions. In both search and optimisation, the positions of the optima are not known in advance (even though the optima itself might be known a-priori). The task of the fitness function is to measure the proximity of the candidate solutions to the optima based on the criteria provided by each optimisation problem. The algorithm compares the output of the function with the output of the previously located candidate solutions and, for instance, in case of a minimisation problem, the smaller the output the better the solution. Data search can be seen as a caste of optimisation if the objective function tests the equality of the candidate solution with the model.

3. Stochastic Diffusion Search

Stochastic Diffusion Search (SDS) [23] introduced a new probabilistic approach for solving best-fit pattern recognition and matching problems. SDS, as a multi-agent population-based global search and optimisation algorithm, is a distributed mode of computation utilising interaction between simple agents [71]. Its computational roots stem from Geoff Hinton's interest 3D object classification and mapping. See [50, 66] for Hinton's work and [23, 25] for the connection between Hinton mapping and SDS.
Unlike many natured inspired search algorithms, SDS has a strong mathematical framework, which describes the behaviour of the algorithm by investigating its resource allocation [81], convergence to global optimum [82], robustness and minimal convergence criteria [79] and linear time complexity [84].
In order to introduce SDS, a social metaphor, *the Mining Game*, is introduced.

3.1. The Mining Game

The mining game provides a simple metaphor outlining the high-level behaviour of agents in SDS:

> A group of friends (miners) learn that there is gold to be found on the hills of a mountain range but have no information regarding its distribution. On their maps the mountain range is divided into a set of discrete hills and each hill contains a discrete set of seams to mine. Over time, on any day the probability of finding gold at a seam is proportional to its net wealth.
>
> To maximise their collective wealth, the miners need to identify the hill with the richest seams of gold so that the maximum number of miners can dig there (this information is not available a-priori). In order to solve this problem, the miners decide to employ a simple Stochastic Diffusion Search.
>
> - At the start of the mining process each miner is randomly allocated a hill to mine (his hill hypothesis, h).
> - Every day each miner is allocated a randomly selected seam on his hill to mine.
> - At the end of each day, the probability that a miner is happy is proportional to the amount of gold he has found.
> - At the end of the day the miners congregate and over the evening each miner who is unhappy selects another miner at random to talk to. If the chosen miner is happy, he happily tells his colleague the identity of the hill he is mining (that is, he communicates his hill hypothesis, h, which thus both now maintain). Conversely, if the chosen miner is unhappy he says nothing and the original miner is once more reduced to selecting a new hypothesis - identifying the hill he is to mine the next day - at random.

In the context of SDS, agents take the role of miners; active agents being 'happy miners', inactive agents being 'unhappy miners and the agent's hypothesis being the miner's 'hill-hypothesis'. It can be shown that this process is isomorphic to SDS, and thus that the miners will naturally self-organise and rapidly congregate over hill(s) on the mountain range with a high concentration of gold.

Algorithm 1 The Mining Game

```
Initialisation phase
Allocate each miner (agent) to a random
       hill (hypothesis) to pick a region randomly

Until (all miners congregate over the highest
       concentration of gold)

   Test phase
     - Each miner evaluates the amount of gold
       they have mined (hypotheses evaluation)
     - Miners are classified into happy (active)
       and unhappy (inactive) groups

   Diffusion phase
     - Unhappy miners consider a new hill by
       either communicating with another miner;
       or,if the selected miner is also
       unhappy, there will be no information
       flow between the miners; instead the
       selecting miner must consider another
       hill (new hypothesis) at random
End
```

3.2. Refinements in the Metaphor

There are some refinements in the miners analogy, which will elaborate more the correlation between the metaphor and different implementations of the algorithm.
Whether an agent is active or not can be measured probabilistically or gold may be considered as resource of discrete units. In both cases all

the agents are either active or inactive at the end of each iteration[2]; this is isomorphic to standard SDS. The Mining Game can be further refined through either of the following two assumptions at each location:

1. Finite resources: the amount of gold is reduced each time a miner mines the area

2. Infinite resources: a conceptual situation with potentially infinite amount of gold

In the case of having finite resources, the analogy can be related to a real world experiment of robots looking for food to return to a notional nest site [62]. Hence the amount of food (or gold, in the mining analogy) is reduced after each discovery. In that experiement, an ant-like algorithm is used to avoid robots interfering with one another; considering individual variation in performing the task; and also recruiting other robots when identifying a rich area is investigated. In this case, the goal is identifying the location of the resources throughout the search space. This type of search is similar to conducting a search in a dynamically, agent-initiated changing environment where agents change their congregation from one area to another.

The second assumption has similarities with discrete function optimisation where values at certain points are evaluated. However further re-evaluation of the same points does not change their values and they remain constant.

3.3. Mathematical Framework

Stochastic diffusion search, unlike many other swarm and evolutionary techniques, has a strong mathematical framework analysing its behaviour in linear time complexity, convergence to global optimum and robustness and minimal convergence criteria.

It has been shown in [84] that SDS can be modelled as an ergodic, finite state Markov Chain under some non-restrictive assumptions. Sub-linear time complexity for some settings of parameters has been formulated and proved; the work shows that in the presence of the data model in a noiseless search space, the SDS algorithm is time sub-linear with the search space size for spaces greater than a supercritical space size and roughly time constant for spaces smaller than supercritical.

The convergence of SDS is proven mathematically in [82], where SDS converges to a statistical equilibrium when it locates the best instantiation of the object in the search space. In other words, it is shown that if the target exists in the search space, all agents will eventually converge to its position. Additionally the notion of convergence is extended in the case when there is no ideal instantiation of the target in the search space and it is proven that convergence also occurs in this case (see Appendix A for more details).

The minimum convergence criteria of SDS is discussed in [79] where an analysis of SDS is presented, leading to a derivation of the minimum acceptable match resulting in a stable convergence within a noisy search space. Therefore, a novel formulation for the SDS algorithm is presented that allows the calculation of the minimum match in a given search space that will guarantee stable convergence of SDS.

[2] Whether an agent is active or not is defined using the following two methods:

- probabilistically:
  a function f takes a probability p as input and returns either true or false, $f(p) \Longrightarrow Active|Inactive$
- discretely:
  if there is gold, the agent will be active, otherwise it will be inactive.

4. SDS Architecture

The SDS algorithm commences a search or optimisation by initialising its population (e.g. miners, in the mining game metaphor). In any SDS search, each agent maintains a hypothesis, h, defining a possible problem solution. In the mining game analogy, agent hypothesis identifies a hill. After initialisation two phases are followed (see Algorithm 1 for these phases in the mining game; for high-level SDS description see Algorithm 2):

- Test Phase (e.g. testing gold availability)
- Diffusion Phase (e.g. congregation and exchanging of information)

In the test phase, SDS checks whether the agent hypothesis is successful or not by performing a partial hypothesis evaluation and returning a domain independent boolean value. Later in the iteration, contingent on the strategy employed, successful hypotheses diffuse across the population and in this way information on potentially good solutions spreads throughout the entire population of agents.

In the Test phase, each agent performs partial function evaluation, pFE, which is some function of the agent's hypothesis; $pFE = f(h)$. In the mining game the partial function evaluation entails mining a random selected region on the hill, which is defined by the agent's hypothesis (instead of mining all regions on that hill).

In the Diffusion phase, each agent recruits another agent for interaction and potential communication of hypothesis. In the mining game metaphor, diffusion is performed by communicating a hill hypothesis.

Algorithm 2 SDS Algorithm

```
01: Initialising agents()
02: While (stopping condition is not met)
03:     Testing hypotheses()
04:         Determining agents' activities (active/inactive)
05:     Diffusing hypotheses()
06:         Exchanging of information
07: End While
```

Although the original SDS model requires full inter-agent connectivity, Section 5.3 describes a lattice implementation, which, while qualitatively retaining the properties of the original algorithm, restricts connectivity, enabling simpler implementation on parallel hardware. Furthermore, details are provided on the diffusion times for different network topologies, ranging from ordered lattices, over small-world networks to random graphs.

4.1. A Search Example

In order to demonstrate the process through which SDS functions, an example is presented which shows how to find a set of letters within a larger string of letters. The goal is to find a 3-letter model (Table 1) in a 16-letter search space (Table 2). In this example, there are four agents. For simplicity of exposition, a perfect match of the model exists in the Search Space (SS).

Table 1. Model

Index:	0	1	2
Model:	***S***	***I***	***B***

Table 2. Search Space

Index:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Search Space	X	Z	A	V	M	Z	S	I	B	V	G	O	L	B	E	H

In this example, a hypothesis, which is a potential problem solution, identifies three adjacent letters in the search space (e.g. hypothesis '1' refers to Z-A-V, hypothesis '10' refers to G-O-L).
In the first step, each agent initially randomly picks a hypothesis from the search space (see Table 3). Assume that:

- the first agent points to the 12^{th} entry of the search space and in order to partially evaluate this entry, it randomly picks one of the letters (e.g. the first one, L): | L | B | E |
- the second agent points to the 9^{th} entry and randomly picks the second letter (G): | V | G | O |
- the third agent refers to the 2^{nd} entry in the search space and randomly picks the first letter (A): | A | V | M |
- the fourth agent goes the 3^{rd} entry and randomly picks the third letter (Z): | V | M | Z |

Table 3. Initialisation and Iteration 1

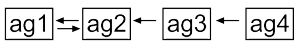
Agent No:	1	2	3	4
Hypothesis position:	12 **L**-B-E	9 V-**G**-O	2 **A**-V-M	3 V-M-**Z**
Letter picked:	1^{st}	2^{nd}	1^{st}	3^{rd}
Status:	×	×	×	×

The letters picked are compared to the corresponding letters in the model, which is S-I-B (see Table 1).
In this case:

- The 1^{st} letter from the first agent (L) is compared against the 1^{st} letter from the model (S) and because they are not the same, the agent is set inactive.
- For the 2^{nd} agent, the second letter (G) is compared with the second letter from the model (I) and again because they are not the same, the agent is set inactive.
- For the third and fourth agents, letters 'A' and 'Z' are compared against 'S' and 'B' from the model. Since none of the letters correspond to the letters in the model, the status of the agents are set inactive.

In the next step, as in the mining game, each inactive agent chooses another agent and adopts the same hypothesis if the selected agent is active. If the selected agent is inactive, the selecting agent generates a random hypothesis.
Assume that the first agent chooses the second one; since the second agent is inactive, the first agent must choose a new random hypothesis from the search space (e.g. 6). See Figure 1 for the communication between agents.

Figure 1. Agents Communication 1



The process is repeated for the other three agents. As the agents are inactive, they all choose new random hypotheses (see Table 4).

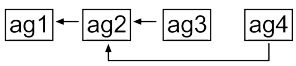
Table 4. Iteration 2

Agent No:	1	2	3	4
Hypothesis position:	6 S-**I**-B	10 G-O-L	0 **X**-Z-A	5 **Z**-S-I
Letter picked:	2^{nd}	3^{rd}	1^{st}	1^{st}
Status:	√	×	×	×

In Table 4, the second, third and fourth agents do not refer to their corresponding letter in the model, therefore they become inactive. The first agent, with hypothesis '6', chooses the 2^{nd} letter (I) and compares it with the 2^{nd} letter of the model (I). Since the letters are the same, the agent becomes active.
At this stage, consider the following communication between the agents: (see Figure 2)

- the fourth agent chooses the second one
- the third agent chooses the second one
- the second agent chooses the first one

Figure 2. Agents Communication 2



In this case, the third and fourth agents, which chose an inactive agent (the second agent), have to choose other random hypotheses each from the search space (e.g. agent three chooses hypothesis '1' which points to Z-A-V and agent four chooses hypothesis 4 which points to M-Z-S), but the second agent adopts the hypothesis of the first agent, which is active. As shown in Table 5:

- The first agent, with hypothesis '6', chooses the 3^{rd} letter (B) and compares it with the 3^{rd} letter of the model (B). Since the letters are the same, the agent remains active.
- The second agent, with hypothesis '6', chooses the 1^{st} letter (S) and compares it with the 1^{st} letter of the model (S). Since the letters are the same, the agent stays active.

· the third and fourth agents do not refer to their corresponding letter in the model, therefore they are set inactive.

Table 5. Iteration 3

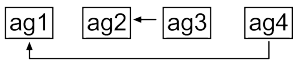
Agent No:	1	2	3	4
Hypothesis position:	6 S-I-**B**	6 **S**-I-B	1 Z-**A**-V	4 M-Z-**S**
Letter picked:	3^{rd}	1^{st}	2^{nd}	3^{rd}
Status:	√	√	×	×

Because the third and fourth agents are inactive, they try to contact other agents randomly. For instance (see Figure 3):

· agent three chooses agent two

· agent four chooses agent one

Figure 3. Agents Communication 3



Since agent three chose an active agent, it adopts its hypothesis (6). As for agent four, because it chose agent one, which is active too, it adopts its hypothesis (6). Table 6 shows:

· The first agent, with hypothesis '6', chooses the 1^{st} letter (S) and compares it with the 1^{st} letter of the model (S). Since the letters are the same, the agent remains active.

· The second agent, with hypothesis '6', chooses the 2^{nd} letter (I) and compares it with the 2^{nd} letter of the model (I). Since the letters are the same, the agent stays active.

· The third agent, with hypothesis '6', chooses the 3^{rd} letter (B) and compares it with the 3^{rd} letter of the model (B). Since the letters are the same, the agent becomes active.

· The fourth agent, with hypothesis '6', chooses the 1^{st} letter (S) and compares it with the 1^{st} letter of the model (S). Since the letters are the same, the agent is set active.

Table 6. Iteration 4

Agent No:	1	2	3	4
Hypothesis position:	6 **S**-I-B	6 S-**I**-B	6 S-I-**B**	6 **S**-I-B
Letter picked:	1^{st}	2^{nd}	3^{rd}	1^{st}
Status:	√	√	√	√

At this stage, the entire agent populations are active pointing to the location of the model inside the search space.

4.2. Initialisation and Termination

Although normally agents are uniformly distributed throughout the search space, if the search space is of a specific type, or knowledge exists about it ***a priori***, it is possible to use a more intelligent (than uniform random distribution) startup by biasing the initialisation of the agents.

If there is a pre-defined pattern to find in the search space, the goal will be locating the best match or, if it does not exist, its best instantiation in the search space [82]. Similarly, in a situation which lacks a pre-defined pattern, the goal will be finding the best pattern in accord with the objective function.

In both cases, it is necessary to have a termination strategy. In one method[3], SDS terminates the process when a statistical equilibrium state is reached, which means that the threshold of the number of active agents is exceeded and the population maintained the same state for a specified number of iterations. In [69], four broad types of halting criteria are introduced:

1. No stopping criterion, where the user interrupts the course of action of the search or optimisation and is usually preferred when dealing with ***dynamically changing*** problem spaces or when there is no predefined pattern to look for

2. Time-based criterion, in which passing a pre-set duration of time is the termination point of the algorithm

3. Activity-based criterion, which is a problem-dependent halting criterion, and it is the most prevalent form in the SDS algorithm. The termination of the process is decided upon through monitoring the overall activity of the agents (e.g. reaching a certain user defined activity level, reaching a stable population state after a sudden increase in their activities)

4. Cluster-based criterion that keeps tracks of the formation of stable clusters.

Determining the termination criteria without having a fixed a priori threshold as a prerequisite is a possible approach (e.g. Quorum sensing, which is a system of stimulus and response correlated to population density, is used in some social insects in search for nest sites or many species of bacteria to coordinate gene expression based on the density of their local population [73]). By the same analogy and as stated in [69], it is possible to implement the termination criterion as a random sampling process: for example, a cluster-based termination procedure may consider monitoring the hypotheses of a subset of the population until the same hypothesis is encountered more than once. This provides partial evidence of the formation of a cluster. The size of the sample taken from the population can be the increased. Such a random sampling procedure could eventually be translated into an SDS algorithm itself: a small number of stopping agents compares the hypotheses of pairs of the original searching agents; the activity state of these stopping agents should be based on the similarity of the two hypotheses. A cluster-based criterion for a rather large population can then be translated into an activity-based criterion for a much smaller population. In some situations, these two populations can even be merged (e.g. active context-sensitive agents already compare hypotheses). A small number of agents can thus simply be designated as 'stopping agents', and be given an extra activity state. In order to avoid

[3] Ibid

adding extra computational cost, the termination procedure does not need to operate in the same time frame as the SDS agents (e.g. it is possible to execute the termination procedure only every n iterations, or every s seconds of computational time).
The two most common termination strategies in SDS (introduced in [82]) are the following:

- Weak halting criterion is the ratio of the active agents to the total number of agents. In this criterion, cluster sizes are not the main concern.
- Strong halting criterion investigates the number of active agents that forms the largest cluster of agents all adopting the same hypothesis.

Therefore, the choice of the halting mechanism is based on whether to favour the active agents in the whole of the agent populations (weak halting mechanism), which is similar to the activity-based criterion, or to consider the largest cluster of active agents (strong halting mechanism), which is similar to the cluster-based criterion.

4.3. Partial Function Evaluation

One of the concerns associated with many optimisation algorithms (e.g. Genetic Algorithm [46], Particle Swarm Optimisation [57] and etc.) is the repetitive evaluation of a computationally expensive fitness functions. In some applications, such as tracking a rapidly moving object, the repetitive function evaluation significantly increases the computational cost of the algorithm. Therefore, in addition to reducing the number of function evaluations, other measures should be taken in order to reduce the computations carried out during the evaluation of each possible solution as part of the optimisation or search processes.
The commonly used benchmarks for evaluating the performance of swarm intelligence algorithms are typically small in terms of their objective functions computational costs [37, 98], which is often not the case in real-world applications. Examples of costly evaluation functions are seismic data interpretation [98], selection of sites for the transmission infrastructure of wireless communication networks and radio wave propagation calculations of one site [97] and etc.
Costly functions have been investigated under different conditions [54] and the following two broad approaches have been proposed to reduce the cost of function evaluations:

- The first is to estimate the fitness by taking into account the fitness of the neighbouring elements, the former generations or the fitness of the same element through statistical techniques introduced in [20, 29].
- In the second approach, the costly fitness function is substituted with a cheaper, approximate fitness function.

When agents are about to converge, the original fitness function can be used for evaluation to check the validity of the convergence [54].
Many fitness functions are decomposable to components that can be evaluated separately. In partial evaluation of the fitness function in SDS, the evaluation of one or more of the components may provide partial information and means for guiding the optimisation.
The partial function evaluation of SDS allows the algorithm to absorb certain types of noise in the objective function without affecting the convergence time or the size and stability of the clusters.
Additionally, noise, which does not alter the averaged probabilities of the test score (probability of producing active agents during the test phase, averaged over all component functions) but increases the variance in the evaluation of component functions, has no effect on the resource allocation process of SDS [72]. However, if the value of test score changes as a result of noise presence, the resource allocation process may be influenced either:

- positively if the value of the test score increases
- or negatively if the value of the test score decreases

4.3.1. Dynamic Environments

The application of partial function evaluation is of more significance when the problem space is dynamically changing and the evaluation process is of more repetitive nature. Repeated (re)evaluations of fitness functions in many swarm intelligence algorithms necessitate having less costly fitness functions.
Diffusion or the selection mechanism tends to reduce the diversity in the population or the population homogeneity [72], which in turn leads to an inadequate subsequent reactions in a dynamically changing fitness function.
SDS aims at proposing a new solution (see Section 4.5.1) to the problem of population homogeneity by utilising an alternative method to balance the trade off between ***wide exploration*** of all possible solution in the problem space and the ***detailed exploitation*** of any possible smaller region which might be a candidate for holding the sought object.

4.4. Convergence

Convergence time is defined as the number of iterations needed before a stable population of active agents is formed.
The SDS algorithm allocates its resources by defining convergence as locating the best match in the search space.
An important factor in convergence is the ratio of the number of agents to the size of the solution space. In [25], it is proved that in a noiseless environment convergence always happens.
Additionally, in [82] it is proved that all agents become active when searching for a solution in a noiseless environment where a perfect match exists.
As mentioned before, the probability of an agent being active averaged over all component functions is the test score, which in turn determines the behaviour of SDS, and it is proved that the population size and the test score determine the average cluster size as well as convergence times.
The approximately linear time complexity of SDS is analysed in [82] and two extereme cases in the convergence time have been considered there:

- First, when, in the initial stages, some of the agents point to the correct position in the search space, which results in a shorter convergence time
- In the second case, there is no agent pointing to the correct position for some time after the initialisation, which may lead to a longer process before the first agent locates a potentially correct location.

It has also been shown that the accuracy and convergence time in SDS is proportionately robust to the amount of noise in the search space.
Convergence to a global optimal solution in SDS is discussed in [84].

4.5. Resource Allocation and Stability

In addition to convergence time, steady-state resource allocation is one of the important factors in the performance criteria of SDS [70]. In order

to measure the robustness of the algorithm, in case of the presence of noise and imperfect matches, resource allocation is taken into account, which is determined as the average number of active states when the search shows steady-state behaviour. Although, resource allocation in standard SDS is *dynamic* and *self-regulatory*, there are certain issues to be investigated.

4.5.1. Local Exploitation and Global Exploration

In standard SDS, there is no explicit mechanism to shift the balance from local exploitation (*detailed exploitation*) to global exploration (*wide exploration*) of candidate solutions.
As observed in [33], a metaheuristic tries to exploit self-similarity and regularities of the fitness function, which indicates that neighbouring solutions in the problem space have alike properties. Adding this mechanism to SDS may be helpful; one way of embedding this into the algorithm is to add a small random offset to the hypotheses before copying them to other agents during the diffusion phase, which is similar to mutation in evolutionary algorithms [69, 72]. The effect of this minor change in the hypotheses is to investigate the nearby solutions, which generally serves as a hill-climbing mechanism improving the overall performance of the SDS and results in improved convergence time in solution spaces with self-similarity. Nevertheless, it also accelerates the finding of more optimal solutions in the vicinity of already found ones.
In dynamically changing environments, it is important to explore the solution space even after finding a suitable candidate solution, as once a good solution is detected, a large propertion of agents are attracted to it, thus limiting further exploration of the solution space. Therefore, the Context Sensitive and Context Free mechanisms (described in Section 5.1) are proposed to shift the balance of the search back to exploration.
A full account of Markov chain based analysis of the stochastic nature of standard SDS for resource allocation and the steady state probability distribution of the whole swarm is extensively discussed in [81]. More information about search behaviour and resource allocation can also be found in [67, 83].
In heuristic multi-agent systems, the possibility of agents losing the best solution results in destabilising or even non-convergence of the algorithm. Conversely, it is shown that the solution found by SDS are exceptionally stable [80].

5. Variations in SDS

In SDS, similar to other optimisation algorithms, the goal is finding the best solution based on the criteria specified in the objective function. The collection of all candidate solutions (hypotheses) forms the search space and each point in the search space is represented by an objective value, from which the objective function is formed [72]. In the minimisation mode, for example, the lower the objective value is the better the result is.
Although there might not be a direct way of finding the best objective function for a problem, many optimisation problems can be transformed into the minimisation form [69].
One of the issues related to SDS is the mechanism behind allocating resources to ensure that while potential areas of the problem space are exploited, exploration is not ignored. For this purpose, different recruitments methods, where one agent recruits another one, are investaged:

5.1. Recruitment Strategies

Three recruitment strategies are proposed in [78]: active, passive and dual. These strategies are used in the Diffusion Phase of SDS. Each agent can be in either one of the following states: It is *active* if the agent is successful in the Test Phase; an agent is *inactive* if it is not successful; and it is *engaged* if it is involved in a communication with another agent.
The standard SDS algorithm [23] uses the passive recruitment mode, which will be described next followed by other recruitment modes.

5.1.1. Passive Recruitment Mode

In the *passive recruitment mode*, if the agent is not active, another agent is randomly selected and if the randomly selected agent is active, the hypothesis of the active agent is communicated (or *diffused*) to the inactive one. Otherwise a new random hypothesis is generated for the inactive agent and there will be no flow of information (see Algorithm 3).

Algorithm 3 Passive Recruitment Mode

```
01: For ag = 1 to No_of_agents
02:     If ( !ag.activity() )
03:         r_ag = pick a random agent()
04:         If ( r_ag.activity() )
05:             ag.setHypothesis( r_ag.getHypothesis() )
06:         Else
07:             ag.setHypothesis( randomHypothesis() )
08:         End If/Else
09:     End If
10: End For
```

5.1.2. Active Recruitment Mode

In the *active recruitment mode*, active agents are in charge of communication with other agents. An active agent randomly selects another agent. If the randomly selected agent is neither active nor engaged in communication with another active agent, then the hypothesis of the active agent is communicated to the inactive one and the agent is flagged as engaged. The same process is repeated for the rest of the active agents. However if an agent is neither active nor engaged, a new random hypothesis is generated for it (see Algorithm 4).

Algorithm 4 Active Recruitment Mode

```
01: For ag = 1 to No_of_agents
02:     If ( ag.activity() )
03:         r_ag = pick a random agent()
04:         If ( !r_ag.activity() AND !r_ag.getEngaged() )
05:             r_ag.setHypothesis( ag.getHypothesis() )
06             r_ag.setEngaged(true)
07:         End If
08:     End If
09: End For
10:
11: For ag = 1 to No_of_agents
12:     If ( !ag.activity() AND !ag.getEngaged() )
13:         ag.setHypothesis( randomHypothesis() )
14:     End If
15: End For
```

5.1.3. Dual Recruitment Mode

In *dual recruitment mode*, both active and inactive agents randomly select other agents. When an agent is active, another agent is randomly selected. If the randomly selected agent is neither active nor engaged, then the hypothesis of the active agent is shared with the inactive one and the inactive agent is flagged as engaged. Also, if there is an agent which is neither active nor engaged, it selects another agent randomly. If the newly selected agent is active, there will be a flow of information from the active agent to the inactive one and the inactive

agent is flagged as engaged. Nevertheless, if there remains an agent that is neither active nor engaged, a new random hypothesis is chosen for it.

Algorithm 5 Dual Recruitment Mode

```
01: For ag = 1 to No_of_agents
02:     If ( ag.activity() )
03:         r_ag = pick a random agent()
04:         If ( !r_ag.activity() AND !r_ag.getEngaged() )
05:             r_ag.setHypothesis( ag.getHypothesis() )
06              r_ag.setEngaged(true)
07:         End If
08:     Else
09:         r_ag = pick a random agent ()
10:         If ( r_ag . activity () AND ! ag . getEngaged () )
11:             ag . setHypothesis ( r_ag . getHypothesis () )
12:             ag . setEngaged ( true )
13:         End If
14:     End If/Else
15: End For
16:
17: For ag = 1 to No_of_agents
18:     If ( !ag.activity() AND !ag.getEngaged() )
19:         ag.setHypothesis( randomHypothesis() )
20:     End If
21: End For
```

5.1.4. Context Sensitive Mechanism

Comparing the above-mentioned recruitment modes, it is theoretically determined in [78] that robustness and greediness decrease in the active recruitment mode. Conversely, these two properties are increased in dual recruitment strategy. Although, the greediness of dual recruitment mode results in decreasing the robustness of the algorithm, the use of *Context Sensitive Mechanism* limits this decrease [78, 81]. In other words, the use of context sensitive mechanism biases the search towards global exploration. In the context sensitive mechanism if an active agent randomly chooses another active agent that maintains the same hypothesis, the selecting agent is set inactive and adopts a random hypothesis. This mechanism frees up some of the resources in order to have a wider exploration throughout the search space as well preventing cluster size from overgrowing, while ensuring the formation of large clusters in case there exists a perfect match or good sub-optimal solutions (see Algorithm 6).

Algorithm 6 Context Sensitive Mechanism

```
01: If ( ag.activity() )
02:     r_ag = pick a random agent ()
03:     If ( r_ag.activity() AND
04:           ag.getHypothsis() == r_ag.getHypothsis() )
05:         ag.setActivity ( false )
06:         ag.setHypotheis ( randomHypothsis() )
07:     End If
08: End If
```

5.1.5. Context Free Mechanism

In *Context Free Mechanism,* which is another recruitment mechanism, the performance is similar to context sensitive mechanism, where each active agent randomly chooses another agent. However, if the selected agent is active (irrespective of having the same hypothesis or not), the selecting agent becomes inactive and picks a new random hypothesis. By the same token, this mechanism ensures that even if one or more good solutions exist, about half of the agents explore the problem space and investigate other possible solutions (see Algorithm 7).

Algorithm 7 Context Free Mechanism

```
01: If ( ag.activity() )
02:     r_ag = pick a random agent ()
03:     If ( r_ag.activity() )
04:         ag.setActivity ( false )
05:         ag.setHypotheis ( randomHypothsis() )
06:     End If
07: End If
```

5.2. Synchronous and Asynchronous Update

Although, in the original SDS [23], synchronous mode is used, the diffusion of successful hypotheses can be accomplished synchronously or asynchronously.

In synchronous diffusion mode, the updates of all hypotheses occur simultaneously (all agents progress through the cycle of test-diffusion at the same time).

There are two methods for asynchronous mode; in the first method, the hypothesis of each agent is updated individually (agents, in turn, go through a test-diffusion cycle). In the second method, there is no explicit synchronisation between agents, which is the case in a true parallel implementation.

As mentioned in [69], in many variants, the behaviour of a asynchronous process is approximately the same as the synchronous one.

5.3. Implementation on Hardware

SDS is inherently parallel in nature and the hardware implementation of the algorithm is feasible. Still, the fact that the original SDS model requires full inter-agent connectivity, where each agent is able to communicate directly with all others in the population, casues fundamental difficulty in the efficient implementation of the algorithm on parallel computer or dedicated hardware.

One of the solutions proposed in [70] was to limit the communication between the agents. Agents are considered spatially located in a lattice (Lattice SDS or LSDS) where each agent is only connected to the k-nearest neighbours.

As a second solution, the agent swarm can be divided into several subswarms. In this mode, each sub-swarm runs on a separate processor and they are fully connected while allowing just a low frequency of communication between swarms. This process is applied to the diffusion phase, during which agents communicate with each another.

Therefore, considering this form of SDS, agents just communicate with the ones they are connected to. It was shown that a network with randomly connected agents (random graph), with small number of long-range connections, performs similar to standard SDS or ordered lattice with roughly the same number of connections[4]. The following conclusion has been drawn that restricting the number of interconnectivity in random or small-world networks – which is a lattice with a few additional number of long-range connections – does not have huge effect on the performance of SDS algorithm. Also, the rate of information spread is higher in random graphs and small-world networks than ordered lattices.

Analysing the number of connections and the connection topology leads to the following conclusion: it has been argued that when a high-dimensional problem is considered, the time at which one of the agents becomes active (time to hit [22]), T_h, is bigger than the time required for

[4] Ibid

the active agent to spread its successful hypothesis T_d [70]. Although random graphs have shorter T_d than regular lattices, they are harder to implement on parallel hardware, because the connection are not necessarily local or short. In small-world lattice SDS topology, which shows the performance of a fully interconnected standard SDS, adding random links decrease T_d exponentially.
Therefore T_d is considered to be an important factor, which not only affect the convergence time, but is also seen as a parameter for resource allocation stability [68] as well as an indirect measure for robustness [70].

5.4. Composite Hypotheses

In standard SDS all hypotheses are homogeneous and thus have the same type. In this section, new variants of SDS are introduced where there are two *different* types of hypotheses working together. These SDS types are applied to solve parameter estimation problems, which is a more complicated search problem compared to pattern matching. In parameter estimation, outlier data (or random noise) is embeded in the data (or inlier data); and the goal is to find parameter values that best describe the inlier data [24]. Data Driven SDS [77] and coupled SDS [24], which have composite hypotheses, are both used to solve parameter estimation problems. An example of parameter estimation problem is locating a spoken word in an audio file which has some noise. In estimation problem, similar to other search problems, a cost function or objective funciton is required to measure how close the algorithm is to the inlier data or the model in the search space.
In parameter estimation, the objective function is optimised with respect to the estimated model parameters; that is why it is considered an optimisation problem [77].

5.4.1. Data Driven SDS

Data Driven SDS (DDSDS) is shown to outperform [77] Maximum Likelihood Estimator Sample Concensus (MLSESAC) which is a variant of RANdom SAmple Consensus (RANSAC), one of the most popular and robust estimators based on stochastic principles [43].
DDSDS contains a composite hypothesis: a manifold hypothesis, which maintains the minimum necessary dataset for describing a hypothesis; and a datum hypothesis, which represent the smallest building block of the hypothesis. If estimating a line is the problem, then the manifold hypothesis would consists of two points, which are sufficient to represent a line, and the datum hypothesis would be a single point that is randomly selected from the manifold hypothesis rather than the whole of the search space.
In the test phase, random datums are selected just from datum hypotheses that are associated with the agents. The probability of selecting a datum, which has no link with any agents is zero. This will dynamically constrain the selection to data generated by the inlier distribution [77]. Next, the distance of the agent's manifold hypothesis from the randomly selected datum is evaluated to see if the distance stays within the pre-set inlier threshold value. If this is the case, the agent's state becomes active.
In the diffusion phase, active agents diffuses its manifold and datum hypotheses to the inactive agent. When an inactive agent is not involved in any information exchange, similar to the initialisation phase, it chooses two random data from the entire search space for the manifold hypothesis and the datum hypothesis is selected from one of the two elements of the manifold hypothesis.

5.4.2. Coupled SDS

In Coupled SDS (CSDS) two independent populations of agents are formed each maintaining different type of hypothesis, namely the manifold hypotheses and datum hypotheses. On the contrary to DDSDS, datum hypotheses are selected randomly from the entire search space. The size of these two populations are not necessarily the same. They are randomly and independently initiated with data from the entire search space. In the test phase, the manifold hypothesis of one agent is compared with the datum hypothesis of another one. Based on the distance threshold, if the datum matches the manifold, both of the agents become active. This evaluation is called composite hypothesis evaluation, which is more complicated than the synchronous evaluation in standard SDS, where there is just one population of agents. Therefore, in addition to asynchronous test, two other synchronisation modes were proposed:

- Master/Slave Synchronisation, where one of the populations is master and the other is slave. The master hypothesis randomly select a hypothesis from the slave population for the test. In this mode, there will be m composite evaluation, where m is the size of the master population.

- Sequential Master Synchronisation is a variant of master/slave mode, where populations take turn to be master. Each iteration has n composite evaluations, which is the sum of all agents in both manifold and datum populations.

The diffusion phase in CSDS is similar to the standard SDS for each population independently, where the information flow is allowed *within* each population of agents and thus there is no information exchange between the manifold and datum population of agents [24].

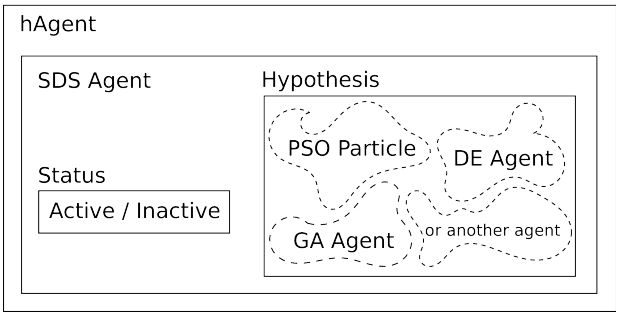
It is empirically shown that DDSDS converges even when there are 50% more outliers and it also outperforms standard SDS in convergence time [77]. Both of these SDS variants have been proposed to improve the performance of the original SDS towards stable convergence in high noise estimation tasks.

Further applications of SDS falling within the categories of continuous optimisation, implementation on hardware, arts and philosophy, and medical applications are presented next.

5.5. Continuous Optimisation

In recent years, SDS algorithm has been deployed for continuous optimisation in several independent research. In 2011 [11], an integration strategy of SDS with Particle Swarm Optimisation (PSO) is proposed, offering a statistically significant outperformance compared to PSO alone. The SDS integration framework was then extended to Differential Evolution (DE) and Genetic Algorithms (GA) [2, 13] demonstrating promising results. SDS integration Framework (SDS-Frame) encapsulates the 'guest' evolutionary algorithms (e.g. PSO, DE, GA, etc.) and facilitates the information exchange between the members of the population (see Fig. 4). The test and diffusion phases are triggered after every n of function evaluations, thus utilising SDS primarily as an efficient resource allocation and dispensation mechanism responsible for facilitating communication between agents.

Figure 4. Hybrid Agent Structure



In other experiments, SDS is adopted for continuous global optimisation, with other evolutionary algorithms utilised providing local search on convergence. In one such instance [10], the optimisation process is initialised by n number of function evaluations (FEs) performed within the SDS test-diffusion cycle in order to allocate the resources (agents) to the promising areas of the search space; and subsequently pass on the agents' positions to a Differential Evolution (DE) algorithm to resume the optimisation process. Hence SDS is utilised as a global optimiser, with DE providing local search on convergence. In another experiment [87], SDS is adopted for continuous global optimisation, using four benchmarks (each with different required accuracies and different maximum number of FEs allowed). In that experiment, SPSO [34] is utilised providing local search on convergence. Following on the previous experiment, in [88], SDS is utilised in the context of unconstrained continuous optimization; the proposed approach uses concepts from probabilistic algorithms to enhance the performance of SDS (Probabilistic SDS or PSDS). PSDS is tested on 16 benchmark functions (10 dimensional problems with varying termination strategy on each group of benchmarks) and is compared with two methods (Cross-Entropy [89] which is a probabilistic algorithm and a variant of Particle Swarm Optimisation which is a swarm intelligence algorithm) showing promising results.

In important work from 2011, SDS was demonstrated to solve the quadratic knapsack problem [65]. The candidate solutions are estimated by the partial function evaluation and the individuals are produced by quantum computation. In this work it was shown that the SDS method was more effective than particle swarm optimisation and ant colony optimisation algorithms.

5.6. NESTER: a connectionist implementation of SDS

NESTOR – the NEural STochastic diffusion search netwORk – consists of an artificial retina, a layer of fully connected matching neurons and retinotopically organised memory neurons. Matching neurons are fully connected to both retina and memory neurons. The information processed by neurons is time-encoded by a spike train consisting of two qualitatively different parts – a tag determined by the relative ***position*** of the receptor on the artificial retina and a ***feature*** signalled by that receptor. The neurons operate by introducing time delays and acting as spatiotemporal coincidence detectors. NESTOR utilises a temporal coding scheme and a dynamic assembly encoding of the target. Finding the target in the search space results in the onset of time locked activity of the assembly of NESTOR neurons. Different features of the same object are bound by their relevant position in the search space and synchronisation of activity within the assembly follows as a result of this binding. In [85] it was shown that NESTOR implements Stochastic Diffusion Search (SDS).

5.7. Stochastic Diffusion Search applied to Trees (SDST): planning and game playing

In a research conducted in 2013 [95], it has been shown that the SDS Swarm Intelligence paradigm can be successfully deployed to solve problems that are usually addressed via the classical algorithmic approach. In this work – Stochastic Diffusion Search applied to Trees (SDST) – communicating populations of SDS agents have been demonstrated to have the capability to address the problem of forward planning. This has been demonstrated via application to the complex, two-person, zero-sum, finite, deterministic strategy game HEX [30].

In SDST, the use of multiple interacting populations of simple stochastic agents can be compared to the dynamics of interacting populations of social insects (e.g. ants) via the concept of 'meta-population' (a term coined in 1969 by Levins [63]). SDST functions as a decentralised, self-organising system as only local rules of interaction between agents are defined and SDST performs forward-planning as it:

- With enough agents and time asymptotically converges to select the 'best move' in the minimax sense.
- Implements a form of Monte-Carlo Tree Search[5].

In other words, it is demonstrated that SDST plays a strong game of HEX [95], successfully avoiding classical tactical errors (forks etc), with the strength of play being contingent on the number of agents and the amount of time allowed to process each move. Further work developing SDST and characterising its behaviour is ongoing.

6. Applications

SDS was first introduced by a simple text searching algorithm in 1989 [22] demonstrating the use of partial function evaluation technique, by partially evaluating the text to find the model or the best match. Since then there have been many applications where SDS has been successfully applied to various diverse problems. This section provides an overview to these problems.

6.1. Art and philosophy

SDS algorithm has been deployed in several artistic applications and in the context of computational creativity, autonomy and swarm intelligence. In one such work [1, 14], the hybrid SDS-PSO algorithm is used to sketch novel drawings of an input image, exploiting an artistic tension between the local behaviour of the 'birds flocking' – as they seek to follow the input sketch – and the global behaviour of the 'ants foraging' – as they seek to encourage the flock to explore novel regions of the canvas. In [14], the paper concludes by exploring the putative 'creativity' of this hybrid swarm system in the philosophical light of the 'rhizome' and Deleuze's well-known 'Orchid and Wasp' metaphor, offering a detailed

[5] Monte-Carlo Tree Search (MCTS) is a recently proposed search method that combines the precision of tree search with the generality of random sampling. Since 2006, over 200 papers related to MCTS have been published, with applications ranging from computer Go, to constraint satisfaction problems, through reinforcement learning and combinatorial optimisation. MCTS has already had a profound impact on Artificial Intelligence approaches for domains that can be represented as trees of sequential decisions, particularly games and planning problems.

Figure 5. Swarmic Sketch

Inspired by Portrait de Diaghilev et Seligsberg by Picasso.

investigation of the 'creativity' of such systems. The relation between the behaviour of the swarm intelligence techniques used and computation creativity is explored in some related publications [5]. In a similar attempt [9], the novel behaviour of the hybrid algorithm assisted by a mechanism inspired from the behaviour of skeletal muscles activated by motor neurons is reflected through a cooperative attempt to make a drawing.

Following the process of drawings facilitated by the behaviour of the underlying swarms, the idea of 'swarmic' sketches and attention mechanism is proposed [3, 7] (see Fig. 5). In this work, the concept of attention is utilised by adapting SDS to selectively attend to detailed areas of a digital canvas. Once the attention of the swarm is drawn to a certain line within the canvas, the capability of another swarm intelligence algorithm – Particle Swarm Intelligence – is used to produce a 'swarmic sketch' of the attended line. The swarms move throughout the digital canvas in an attempt to satisfy their dynamic roles – attention to areas with more details – associated to them via their fitness function. Having associated the rendering process with the concepts of SDS-led attention, the performance of the participating swarms creates a unique, non-identical sketch each time the 'artist' swarms embark on interpreting the input line drawings. A brief account of the 'computational creativity' of the work is given through two prerequisites of creativity within the swarm intelligence's two infamous phases of exploration and exploitation; these phases are also described through the attention and tracing mechanisms respectively. The concept of SDS-led attention is extendible to other measures such as colour which is explored in another work introducing swarmic paintings [6], where SDS is used for producing non-photorealistic images (see Fig. 6).

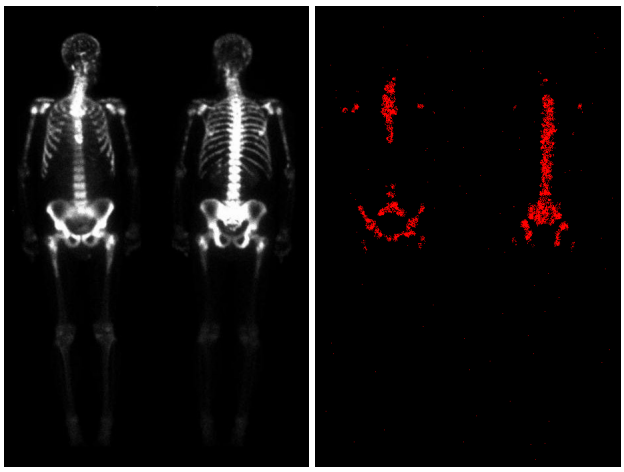
Figure 6. Swarmic Painting

Top: original photo; middle and bottom: snapshots of the images produced.

6.2. Medical applications

Swarm intelligence techniques have offered insightful assistance in many real-world problems, including medical imaging. In the first work of its kind where SDS is deployed to address problems in this field, the goal was to visualise the swarms' movements when presented with a two dimensional canvas representing bone scans [8] (see Fig. 7). This work was well received as a potential educational tool for doctors in training and medical students. This led to the extension of the research in [4] where the application of this swarm intelligence technique on bone scan was introduced in further details in different venues for researchers with medical and computer backgrounds.

Figure 7. Identification of metastasis in bone scan



Later in [12], the statistical and mathematical models were introduced for bone scans, and the application of the technique was extended to mammography. Ongoing work includes analysing CT scans for detecting and highlighting any possible calcifications around the aorta with the goal of assisting the radiologists to determine the extent of the calcification, as well as the identification of the tip of the Nasogastric tube in chest X-rays [15] (see Fig 8).

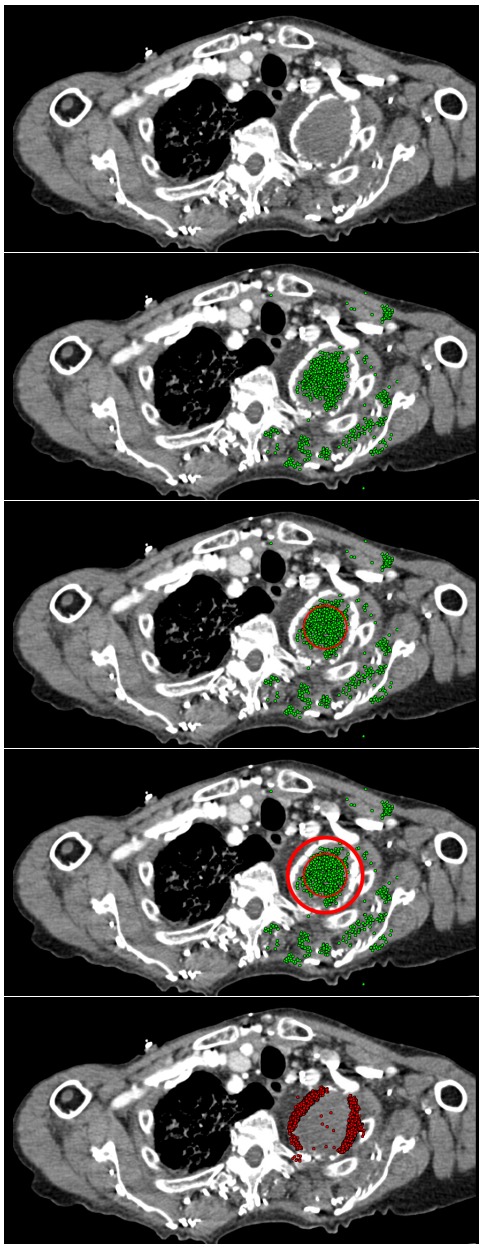
6.3. Other applications

Since its inception in 1989 [22] there have been many other notable applications of SDS; these include:

- **Eye tracking:** in 1992, tracking eyes was investigated in [25]. In this project, a hybrid stochastic search network was used to locate eye positions within grey scale images of human faces. It was shown that the network can accurately locate the eye features of all the subjects it has been trained with and it could reach over sixty percent success in locating eye features on subjects on which the system has not been explicitly trained with.

- **Lip tracking:** in 1995 SDS was again deployed on another visual search, object recognition task. Here Grech Cini [48] deployed a hybrid system of n-tuple neurons [16] and SDS in real time to locate and track of facial features (e.g. lips) in video.

Figure 8. Identification of calcifications around the aorta



The image on top is the original CT scan and the rest show the process through which calcifications around aorta is identified.

- **Localisation:** in 1998 Beattie and Bishop [19] used a 'Focused Stochastic Diffusion Network' (FSDS) to self-localise autonomous wheelchair robot in a complex busy environment. In this work a high resolution map of the robot's locale was used and data from a 2D laser-scanner correlated with possible positions in the map. FSDS initially quantise the enormous search space of possible robot positions (grid cells on the map) to a

much smaller set of 'course', lower resolution, cells. SDS agents initially operate at the course resolution; if an agent finds evidence that the robot is located in a cell at the course resolution, the agent focusses into the cell at a higher resolution and the process is iterated. FSDS terminates when a population of agents stabilise and maintain the same hypothesis at a suitably high resolution.

- **Wireless networks:** in 2002 SDS was also used in wireless transformation networks, where the location of transmission infrastructure is particularly important in order to keep network costs at a minimum whilst preserving adequate area coverage [97]. In this application, given a set of candidate sites, SDS was used to help design network structure so that at required reception points on the network, the signal from at least one transmitter can be received.

- **Sequence detection:** a version of SDS - constrained Stochastic Diffusion Search (CSDS) [56] - was first used to detect partial sequences of strings in 2002. Constrained SDS is an extension to best-fit string-matching SDS while allowing the identification of best-fit sequences (usually referred to as optimal alignment[96]), where there might be gaps between contiguous sub-strings of a model in the search space. CSDS has application to the field of computational molecular biology (e.g. identifying regions of DNA that would code for an amino-acid sequence).

- **Head tracking:** in 2005, using Group Stochastic Search (GSS) [41] was applied to a another visual tracking problem, this application investigated the possibility of locating and tracking objects (e.g. head) in cluttered environment. In this application, each agent utilises SDS, an n-tuple weightless neural network [76] and a histogram intersection technique to score its location [21]. Since the application works when the speed is high, exhaustive and computationally expensive searches for the head are not practical. GSS is an extension for video of Summers work [94] which was introduced to located known patterns in images but which was not designed to operate on the changing search space of real-time video.

- **Voting systems:** in 2006 [86], SDS was deployed in the context of voting methods, where performance comparison to match a correct ordering of a number of voting algorithms that are derived from known social choice rules are made. In this research a standard version of SDS algorithm was adapted for this purpose.

- **Feature tracking:** in 2008 [49], SDS was used in feature tracking in the context of the derivation of Atmospheric Motion Vectors, as conventional template matching techniques, (such as Euclidean distance or cross-correlation for tracking steps) was too very expensive computationally.

- **Room design:** in 2009, among various heuristic methods, SDS is used [31] within a general scheme for the automatic creation of virtual rooms. Certain combinations of objects and their likely location (e.g. books typically being found on a bookshelf) allow very simple placement methods to be used, whilst most general situations were addressed using Stochastic Diffusion Search.

- **Advertising:** the concept of using SDS in social networks (in conjunction with concepts from economics) was investigated in 2010 [90]. The resulting algorithm was termed 'Stochastic Diffusion Market Search' (SDMS). In SDMS a novel contextual advertising method for mutual advertisement hosting amongst participating entities (each owning a website) is proposed. In the suggested method the advertising market and network that formed in the system emerge from agent's preferences and their social behaviour in the network. It was shown that a SDMS network potentially converges to a stable stage and at convergence the distribution of market prices adheres to power-law properties.

- **Cellular automata:** in 2011 research investigated the interplay of three nature inspired techniques: cellular automata (with origins in crystalline lattice formation) were coupled with an artificial immune system (using the clonal selection principle) in order to regulate the convergence of the stochastic diffusion search algorithm [35]. The work primarily investigated the role of the cellular automata (of differing complexity classes) in establishing a balancing mechanism between exploitation and exploration in the emergent behaviour of the system.

- **Reinforcement learning:** the use of stochastic diffusion search with reinforcement learning was recently explored [53]. In this work, it was demonstrated that the application of SDS was able to discover the majority of instances of strong correlations between artificially generated time series at different time indexes.

- **NP-Hard problems:** in 2012 SDS was applied to the rectilinear Steiner minimum tree problem [64]; this problem requires the derivation of the shortest tree connecting given points in the plane using rectilinear distance; it has extensive applications in many real world problems and is known to be NP-hard. A cellular automata based stochastic diffusion search algorithm was used to solve the rectilinear Steiner minimum tree problem, as it exhibited low time complexity. Experimental evidence demonstrated that the algorithm also works well in practice even for a large scale rectilinear Steiner minimum tree.

7. Conclusions

This paper gives a brief account of the research carried out on stochastic diffusion search, a population-based, nature-inspired probabilistic approach, which solves best-match problems mainly by communication between agents. An important feature that makes SDS different from many other optimisation techniques is the mathematical framework that proves its convergence to optimal solution even in noisy search spaces and the stability of the solutions it finds.

SDS gains computational traction via two mechanisms - its *partial function evaluation* capability which leverages particular force when the objective function is both computationally expensive and decomposable; and the rapid positive feedback of potentially good hypotheses between agents.

SDS has been used in dynamically changing search and tracking environments and in contrast to many connectionist models (e.g. those that find the solution by approaching a specific point in the weight space which results in decreasing of their activity after convergence) SDS is naturally able to continue the exploration over the search space further even after locating the optimum.

Medical applications of SDS have also been explored in areas such as detecting metastasis and calcifications in bone scans and mammo-

graphs, as well as the deployment of SDS agents for the identifications of possible calcifications around the aorta in CT scans.
Over the last decade SDS has extended away from its first roots in discrete search and optimisation problems, to be applied in the domain of continuous optimisation. Here it has also been merged with other swarm intelligence optimisers. SDS has also demonstrated promise in applications involving forward planning and it has been successfully demonstrated on an NP-hard problem.
Stochastic diffusion search has also been deployed in several art and philosophy research topics with special emphasis on computational creativity, autonomy and attention.

References

[1] al-Rifaie MM (2011) D-Art 2011: When birds and ants set off to draw. 15th International Conference Information Visualisation (iV2011, London) & 8th International Conference Computer Graphics, Imaging and Visualization (cgiv2011, Singapore) - DIGITAL ART GALLERY
[2] al-Rifaie MM (2012) Information sharing impact of stochastic diffusion search on population-based algorithms. PhD thesis, Goldsmiths, University of London
[3] al-Rifaie MM (2013) D-Art 2013: Swarmic sketches with swarmic attention. 17th International Conference Information Visualisation (iV2013, London, UK) & 10th International Conference Computer Graphics, Imaging and Visualization (cgiv2013, Macau, China) - DIGITAL ART GALLERY
[4] al-Rifaie MM, Aber A (2012) Identifying metastasis in bone scans with stochastic diffusion search. In: Information Technology in Medicine and Education (ITME), IEEE, , URL http://dx.doi.org/10.1109/ITiME.2012.6291355
[5] al-Rifaie MM, Bishop M (2013) Swarm intelligence and weak artificial creativity. In: The Association for the Advancement of Artificial Intelligence (AAAI) 2013: Spring Symposium, Stanford University, Palo Alto, California, U.S.A., pp 14–19
[6] al-Rifaie MM, Bishop M (2013) Swarmic paintings and colour attention. In: Machado P, McDermott J, Carballal A (eds) Evolutionary and Biologically Inspired Music, Sound, Art and Design, Lecture Notes in Computer Science, vol 7834, Springer Berlin Heidelberg, pp 97–108, , URL http://dx.doi.org/10.1007/978-3-642-36955-1_9
[7] al-Rifaie MM, Bishop M (2013) Swarmic sketches and attention mechanism. In: Machado P, McDermott J, Carballal A (eds) Evolutionary and Biologically Inspired Music, Sound, Art and Design, Lecture Notes in Computer Science, vol 7834, Springer Berlin Heidelberg, pp 85–96, , URL http://dx.doi.org/10.1007/978-3-642-36955-1_8
[8] al-Rifaie MM, Aber A, Raisys R (2011) Swarming robots and possible medical applications. In: International Society for the Electronic Arts (ISEA 2011), Istanbul, Turkey
[9] al-Rifaie MM, Bishop M, Aber A (2011) Creative or not? birds and ants draw with muscles. In: AISB 2011: Computing and Philosophy, University of York, York, U.K., pp 23–30, iSBN: 978-1-908187-03-1
[10] al-Rifaie MM, Bishop M, Blackwell T (2011) An investigation into the use of swarm intelligence for an evolutionary algorithm optimisation. In: International Conference on Evolutionary Computation Theory and Application (ECTA 2011), IJCCI
[11] al-Rifaie MM, Bishop MJ, Blackwell T (2011) An investigation into the merger of stochastic diffusion search and particle swarm optimisation. In: Proceedings of the 13th annual conference on Genetic and evolutionary computation, ACM, New York, NY, USA, GECCO '11, pp 37–44, , URL http://doi.acm.org/10.1145/2001576.2001583
[12] al-Rifaie MM, Aber A, Oudah AM (2012) Utilising stochastic diffusion search to identify metastasis in bone scans and microcalcifications on mammographs. In: Bioinformatics and Biomedicine (BIBM 2012), Multiscale Biomedical Imaging Analysis (MBIA2012), IEEE, pp 280–287, URL http://dx.doi.org/10.1109/BIBMW.2012.6470317
[13] al-Rifaie MM, Bishop M, Blackwell T (2012) Information sharing impact of stochastic diffusion search on differential evolution algorithm. In: J. Memetic Computing, vol 4, Springer-Verlag, pp 327–338, , URL http://dx.doi.org/10.1007/s12293-012-0094-y
[14] al-Rifaie MM, Bishop M, Caines S (2012) Creativity and autonomy in swarm intelligence systems. In: J. Cognitive Computation, vol 4, Springer-Verlag, pp 320–331, , URL http://dx.doi.org/10.1007/s12559-012-9130-y
[15] al-Rifaie MM, Aber A, Oudah AM (2013) Ants intelligence framework; identifying traces of cancer. In The House of Commons, UK Parliment. SET for BRITAIN 2013. Poster exhibitions in Biological and Biomedical Science
[16] Aleksander I, Stonham T (1979) Computers and digital techniques 2(1). Lect Notes Art Int 1562 pp 29–40
[17] Ashby W (1960) Design for a Brain. Chapman and Hall London
[18] Back T (1996) Evolutionary Algorithms in Theory and Practice. New York: Oxford University Press
[19] Beattie P, Bishop J (1998) Self-localisation in the senario autonomous wheelchair. Journal of Intellingent and Robotic Systems 22:255–267
[20] el Beltagy MA, Keane AJ (2001) Evolutionary optimization for computationally expensive problems using gaussian processes. In: Proc. Int. Conf. on Artificial Intelligence'01, CSREA Press, pp 708–714
[21] Birchfield S (1998) Elliptical head tracking using intensity gradients and color histograms. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Citeseer, pp 232–237
[22] Bishop J (1989) Anarchic techniques for pattern classification. PhD thesis, University of Reading, Reading, UK
[23] Bishop J (1989) Stochastic searching networks. Proc. 1st IEE Conf. on Artificial Neural Networks, London, UK, pp 329–331
[24] Bishop J (2003) Coupled stochastic diffusion processes. In: Proc. School Conference for Annual Research Projects (SCARP), Reading, UK, pp 185–187
[25] Bishop J, Torr P (1992) The stochastic search network. In: Neural Networks for Images, Speech and Natural Language, Chapman & Hall, New York, pp 370–387
[26] Bishop M, de Meyer K, Nasuto S (2002) Recruiting robots perform stochastic diffusion search. School Conference for Annual Research Projects (SCARP)
[27] Bonabeau E, Dorigo M, Theraulaz G (1999) Swarm intelligence: from natural to artificial systems. Oxford University Press, USA
[28] Bonabeau E, Dorigo M, Theraulaz G (2000) Inspiration for optimization from social insect behaviour. Nature 406:3942
[29] Branke J, Schmidt C, Schmeck H (2001) Efficient fitness estimation in noisy environments. In Spector, L, ed: Genetic and Evolutionary Computation Conference, Morgan Kaufmann
[30] Browne C (2000) Hex Strategy: Making the right connections. AK Peters Wellesley
[31] Cant R, Langensiepen C (2009) Methods for Automated Object Placement in Virtual Scenes. In: UKSim 2009: 11th International Conference on Computer Modelling and Simulation, IEEE, pp 431–436

[32] Chadab R, Rettenmeyer C (1975) Mass recruitment by army ants. Science 188:1124–1125
[33] Christensen S, Oppacher F (2001) What can we learn from no free lunch? a first attempt to characterize the concept of a searchable function. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp 1219–1226
[34] Clerc M (2010) From theory to practice in particle swarm optimization. Handbook of Swarm Intelligence pp 3–36
[35] Coulter D, Ehlers E (2011) Cellular automata and immunity amplified stochastic diffusion search. In: Advances in Practical Multi-Agent Systems, Springer, pp 21–32
[36] Deneubourg J, Pasteels J, Verhaeghe J (1983) Probabilistic behaviour in ants: a strategy of errors? In: Journal of Theoretical Biology, Elsevier, vol 105, pp 259–271
[37] Digalakis J, Margaritis K (2002) An experimental study of benchmarking functions for evolutionary algorithms. International Journal 79:403–416
[38] Dorigo M (1992) Optimization, learning and natural algorithms. PhD thesis, Milano: Politecnico di Italy
[39] Dorigo M, Maniezzo V, Colorni A (1991) Positive feedback as a search strategy. Dipartimento di Elettronica e Informatica, Politecnico di
[40] Dorigo M, Caro GD, Gambardella LM (1999) Ant algorithms for discrete optimization. Artificial Life 5(2):137–172
[41] Evans M, Ferryman J (2005) Group stochastic search for object detection and tracking. Advanced Video and Signal Based Surveillance, 2005 AVSS 2005 IEEE Conference
[42] Fan H, Hua Z, Li J, Yuan D (2004) Solving a shortest path problem by ant algorithm. In: Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on, vol 5, pp 3174–3177 vol.5,
[43] Fischler MA, Bolles RC (1981) Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM 24(6):381–395
[44] Fogel DB (1995) Evolutionary Computation: Toward a New Philosophy of Machine Intelligence. IEEE Press, Piscataway, NJ
[45] Glover F, et al (1989) Tabu search-part i. ORSA journal on Computing 1(3):190–206
[46] Goldberg DE (1989) Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Longman Publishing Co., Inc. Boston, MA, USA
[47] Goodman LJ, Fisher RC (1991) The Behaviour and Physiology of Bees. CAB International, Oxon, UK
[48] Grech-Cini E (1995) Locating facial features. PhD thesis, University of Reading, Reading, UK
[49] Hernandez-Carrascal A, Nasuto S (2008) A swarm intelligence method for feature tracking in amv derivation. Ninth International Wind Workshop
[50] Hinton GF (1981) A parallel computation that assigns canonical object-based frames of reference. In: Proceedings of the 7th international joint conference on Artificial intelligence-Volume 2, Morgan Kaufmann Publishers Inc., pp 683–685
[51] Holland JH (1975) Adaptation in natural and artificial systems. Ann Arbor, MI, University of Michigan press
[52] Holldobler B, Wilson EO (1990) The Ants. Springer-Verlag
[53] Hughes R (2012) Stochastic diffusion search with reinforcement learning. In: Proc. School Conference for Annual Research Projects (SCARP), Reading, UK
[54] Jin Y (2005) A comprehensive survey of fitness approximation in evolutionary computation. In: Soft Computing 9:3–12
[55] Jin Y, Branke J (2005) Evolutionary optimization in uncertain environments-a survey. Evolutionary Computation, IEEE Transactions on 9(3):303–317
[56] Jones D (2002) Constrained stochastic diffusion search. Proc School Conference for Annual Research Projects (SCARP) 2002, Reading, UK
[57] Kennedy J, Eberhart RC (1995) Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks, IEEE Service Center, Piscataway, NJ, vol IV, pp 1942–1948
[58] Kemeny, J.G. & Snell, J.L., (1976), Finite Markov Chains, New York: Springer-Verlag.
[59] Kennedy JF, Eberhart RC, Shi Y (2001) Swarm intelligence. Morgan Kaufmann Publishers, San Francisco ; London
[60] Kirkpatric S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220(4598):671–680
[61] Knuth DE (1973) The art of computer programming. Vol. 3, Sorting and Searching. Addison-Wesley Reading, MA
[62] Krieger MJ, Billeter JB, Keller L (2000) Ant-like task allocation and recruitment in cooperative robots. Nature 406(6799):992–5
[63] Levins R (1969) Some demographic and genetic consequences of environmental heterogeneity for biological control. Bulletin of the ESA 15(3):237–240
[64] Li SW, Zhang J (2012) Cellular sds algorithm for the rectilinear steiner minimum tree. In: Digital Manufacturing and Automation (ICDMA), 2012 Third International Conference on, IEEE, pp 272–276
[65] Liu Y, Ma L (2011) Stochastic diffusion search algorithm for quadratic knapsack problem. Control Theory & Applications 28(8):1140–1144
[66] McClelland JL, Rumelhart DE, Group PR, et al (1986) Parallel distributed processing. Explorations in the microstructure of cognition 2
[67] de Meyer K (2000) Explorations in stochastic diffusion search: Soft- and hardware implementations of biologically inspired spiking neuron stochastic diffusion networks. Tech. Rep. KDM/JMB/2000/1, University of Reading
[68] de Meyer K (2000) Explorations in stochastic diffusion search: Soft-and hardware implementations of biologically inspired spiking neuron stochastic diffusion networks. Tech. rep., Technical Report KDM/JMB/2000
[69] de Meyer K (2003) Foundations of stochastic diffusion search. PhD thesis, PhD thesis, University of Reading, Reading, UK
[70] de Meyer K, Bishop M, Nasuto S (2002) Small world effects in lattice stochastic diffusion search. In: Proc. ICANN 2002, Lecture Notes in Computer Science, 2415, Madrid, Spain, pp 147–152
[71] de Meyer K, Bishop JM, Nasuto SJ (2003) Stochastic diffusion: Using recruitment for search. Evolvability and interaction: evolutionary substrates of communication, signalling, and perception in the dynamics of social complexity (ed P McOwan, K Dautenhahn & CL Nehaniv) Technical Report 393:60–65
[72] de Meyer K, Nasuto S, Bishop J (2006) Stochastic diffusion optimisation: the application of partial function evaluation and stochastic recruitment in swarm intelligence optimisation. Springer Verlag 2, Chapter 12 in Abraham, A. and Grosam, C. and Ramos, V. (eds), "Swarm intelligence and data mining"
[73] Miller MB, Bassler BL (2001) Quorum sensing in bacteria. Annual Reviews in Microbiology 55(1):165–199
[74] Mitchell M (1996) An introduction to genetic algorithms, 1996. MIT press
[75] Moglich M, Maschwitz U, Holldobler B (1974) Tandem calling: A new kind of signal in ant communication. Science 186(4168):1046–1047
[76] Morciniec M, Rohwer R (1995) The n-tuple classifier: Too good to ignore. Tech. Rep. Technical Report NCRG/95/013

[77] Myatt D, Bishop J (2003) Data driven stochastic diffusion networks for robust high-dimensionality manifold estimation - more fun than you can shake a hyperplane at. In: Proc. School Conference for Annual Research Projects (SCARP), Reading, UK
[78] Myatt D, Nasuto S, Bishop J (2006) Alternative recruitment strategies for stochastic diffusion search. Artificial Life X, Bloomington USA
[79] Myatt DR, Bishop JM, Nasuto SJ (2004) Minimum stable convergence criteria for stochastic diffusion search. Electronics Letters 40(2):112–113
[80] Nasuto S, Bishop M (2007) Stabilizing swarm intelligence search via positive feedback resource allocation. In: Nature Inspired Cooperative Strategies for Optimization (NICSO), Springer
[81] Nasuto SJ (1999) Resource allocation analysis of the stochastic diffusion search. PhD thesis, University of Reading, Reading, UK
[82] Nasuto SJ, Bishop JM (1999) Convergence analysis of stochastic diffusion search. Parallel Algorithms and Applications 14(2)
[83] Nasuto SJ, Bishop MJ (2002) Steady state resource allocation analysis of the stochastic diffusion search. Arxiv preprint cs/0202007
[84] Nasuto SJ, Bishop JM, Lauria S (1998) Time complexity analysis of stochastic diffusion search. Neural Computation NC98
[85] Nasuto SJ, Dautenhahn K, Bishop J (1999) Communication as an emergent methaphor for neuronal operation. Lect Notes Art Int 1562 pp 365–380
[86] Nircan A (2006) Stochastic diffusion search and voting methods. PhD thesis, Bogaziki University
[87] Omran M, Moukadem I, al-Sharhan S, Kinawi M (2011) Stochastic diffusion search for continuous global optimization. International Conference on Swarm Intelligence (ICSI 2011), Cergy, France
[88] Omran MG, Salman A (2012) Probabilistic stochastic diffusion search. In: Swarm Intelligence, Springer, pp 300–307
[89] Rubinstein RY, Kroese DP (2004) The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning. Springer Verlag
[90] Salamanos N, Lopatatzidis S, Vazirgiannis M, Thomas A (2010) Advertising network formation based on stochastic diffusion search and market equilibria. In: Proceedings of the 28th ACM International Conference on Design of Communication, ACM, pp 81–87
[91] Saxe JG, Lathen D, Chief B (1882) The Blind Man and the Elephant. The Poems of John Godfrey Saxe
[92] Seeley TD (1995) The Wisdom of the Hive. Harvard University Press
[93] Storn R, Price K (1997) Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. J Global Optim 11:341–359
[94] Summers R (1998) Stochastic diffusion search: A basis for a model of visual attention?
[95] Tanay T, Bishop J, Nasuto S, Roesch E, Spencer M (2013) Stochastic diffusion search applied to trees: a swarm intelligence heuristic performing monte-carlo tree search. In: Proc AISB 2013, University of Exeter, UK
[96] Tompa M (2000) Lecture notes on biological sequence analysis. Dept of Comp Sci and Eng, University of Washington, Seattle, Technical report
[97] Whitaker R, Hurley S (2002) An agent based approach to site selection for wireless networks. In: 1st IEE Conf. on Artificial Neural Networks, ACM Press Proc ACM Symposium on Applied Computing, Madrid Spain
[98] Whitley D, Rana S, Dzubera J, Mathias KE (1996) Evaluating evolutionary algorithms. Artificial Intelligence 85(1-2):245–276

Appendix

A. Proof of SDS convergence to global optimum

The following proof is from Nasuto, S.J. & Bishop, J.M., (1999), *Convergence analysis of Stochastic Diffusion Search*, Parallel Algorithms 14(2), pp. 89-109. Gordon and Breach Publishers [82].

A.1. The model

In the most general case, stochastic diffusion search is supposed to locate the target or if it does not exist in the search space its best instantiation. Therefore from now on we will refer to the object sought by SDS as the target.

Let the search space size be N (measured as a number of possible locations of objects). Let the probability of locating the target in a uniformly random draw be p_m and let the probability of locating the suboptimal object (one sharing to some extent common features with the target) be p_d. Let the probability of a false positive be p^+ and the probability of false negative be p^-. Assume that there are M agents. The state of the search in the n_{th} step is determined by the number of active agents pointing towards the position of the target and active agents pointing towards the false positives (the number of nonactive agents will be equal to the difference between the total number of agents and the two numbers). This is because, by assumption, only active agents are considered as carrying potentially useful information and effectively they influence the search directions of all other agents. Also the strong halting criterion uses only information from active agents.

Thus in effect we have finite number of discrete states each characterised by the pair of two natural numbers. Stochastic Diffusion Search changes its state in a random manner and the possible future evolution of the SDS can be influenced by the past only via the present state (agents are memoryless and the information about the past evolution of SDS is contained in its current configuration) thus effectively it can be modelled by a Markov Chain.

In order to specify the Markov Chain model we will construct the transition matrix. Let the state of the search in the n_{th} step, denoted X_n, be specified by a pair of integers (v, b), where v denotes a number of active agents pointing towards the target and b the number of active agents pointing towards false positives. If in the n_{th} step an agent is active and points towards the target then it will become inactive with probability p^-, otherwise it will remain active. Similarly an active agent pointing towards the false positive will remain active with probability p^+, otherwise it will become inactive.

The one step evolution of the nonactive agent is determined first by the outcome of the diffusion phase and then by the testing phase. We will describe here one of its possible evolutions. During the diffusion phase a nonactive agent will choose an active agent pointing towards the target with probability v/M and then will remain active with probability $1 - p^-$. The other possibilities follow in an analogous way.

It is apparent that transition from one state to another can take place in many different ways depending on the performance of all agents (e.g. number of nonactive agents can increase by one, because during one iteration an active agent pointing towards a false negative failed the test phase or two active agents pointing to the target became inactive and one inactive agent became active and so on). The one step probabilities of transition from one state to another result from summing probabilities of all possible ways that this particular transition can be achieved. The exact formula is given below:

$$P\{X_{n+1} = (r,a)|(X_n = (v,b)\} = \sum_{k_2}^{min(v,r)} \sum_{k_1}^{min(b,a)} Bin(k_2, p^-)Bin(k_1, p^+) Mult(k_1, k_2, r, a, v, b)$$

$$Bin(k_2, p^-) = \binom{v}{k_2}(1-p^-)^{k_2}(p^-)^{v-k_2}$$

$$Bin(k_1, p^+) = \binom{b}{k_1}(p^+)^{k_1}(1-p^+)^{b-k_1}$$

$$Mult(k_1, k_2, r, a, v, b) = \binom{M-v-b}{r-k_2} p_{ab}^{r-k_2} \binom{M-v-b-r+k_2}{a-k_1} \times p_{af}^{a-k_1}(1-p_{ab}-p_{af})^g$$

$$p_{ab} = \frac{v}{M}(1-p^-) + (1-\frac{v}{M}-\frac{b}{M})p_m(1-p^-)$$

$$p_{af} = \frac{b}{M}(p^+) + (1-\frac{v}{M}-\frac{b}{M})p_d p^+$$

.. and the double summation in the above formula is over such $k_1, k_2 \geq 0$, that $g \geq 0$.

The term $Bin(k_2, p^-)$ denotes the probability that k_2 out of v active agents pointing towards the target will remain active after the test phase and $v - k_2$ agents will become inactive. Similarly, the term $Bin(k_1, p^+)$ denotes the probability, that k_1 out of b active agents pointing towards false positives will remain active after testing and $b - k_1$ of them will become nonactive. The term $Mult(k_1, k_2, r, a, v, b)$ expresses the probability of $r - k_2$ out of $M - v - b$ inactive agents starting to point towards the target and passing the test phase, $a - k_1$ out of $M - v - b - r + k_2$ agents starting to point towards false positives and become active and remaining agents staying inactive.

Finally observe that the above formula can be extended for cases when p^-, p^+ are equal to zero by calculating the limit of the transition probability with p^-, p^+ tending to zero respectively.

Let S denote a given search space. Let f_n^s denote the number of active agents pointing to the same position s in the search space S in the n_{th} iteration. It is easy to see that the following condition is fulfilled: $\sum_{s \in S} f_n^s \leq M$, where M is the total number of agents.

Let z_n denote the maximal number of active agents in the n_{th} iteration pointing to the same position, $s_n^z \in S$ in the search space, i.e. $z_n = max_{s \in S}(f_n^s)$. Then, from Bishop [23], the definition of convergence of stochastic diffusion search has the following formulation:

Definition 1. Strong halting criterion. *We say that stochastic diffusion search has reached an equilibrium, if*

$$\exists_{a,b>0}(2b < M \wedge b + a \leq M \wedge a - b \geq 0) \quad (1)$$

$$\exists_{n_0} \forall_{n>n_0}(|z_n - a|) < b) \quad (2)$$

.. and the solution is the position pointed at by z_n.

Thus stochastic diffusion will have reached an equilibrium if there exists a time instant n_0 and an interval (specified by a and b) such that after n_0 the maximal number of agents pointing to the same position will enter and remain within the specified interval. Intuitively, the competitive cooperation process will lead to the allocation of most agents to the best fit position.

Note also, that the above definition does not require convergence of the process to a fixed point. Indeed, the interval specified by a and b defines a tolerance region. All fluctuations of the maximal number of agents pointing to the same position in the search space are discarded as not important, if they occur within this interval. The conditions for a and b exclude the trivial case in which we would ask only, that $0 \leq z_n \leq M$.

It will be shown that in the case of ideal instantiation of the target in the search space these two parameters do not play a critical role. In the opposite case we are faced with the difficult problem. Namely a and b are related to the ability of agents to point towards the best instantiation of the target, (i.e. they are negatively correlated to the probability of false negative) but this is not known in advance in the most general case. The possible solution is for the user of the SDS to assume the minimal acceptance level for the object to be recognised and to estimate suitable values of a and b off-line from this acceptance level.

A.2. Convergence of Stochastic Diffusion Search

We will analyse the convergence of SDS in two separate cases. First we will concentrate on the case when there exist the ideal instantiation of the target in the search space. In the presence of the target in the search space the testing phase for agents pointing to the target becomes deterministic (there is a perfect match, so no agent pointing to this position can fail the test). In what follows we will use the notation introduced in section A.1.

Let the position of the model in the search space be denoted as s_m. Recall that in our Markov chain model of stochastic diffusion the presence of the object in the search space is equivalent to setting p^- to zero.

Proposition 1. *If $p^- = 0$, then*

$$P\{\lim_{n\to\infty} Z_n = M\} = 1 \quad (3)$$

Moreover, $P\{s_n^z = s^m\} = 1$, where $z^n = max_{s \in S}(f_n^s)$.

Proof. From transition probability matrix and from $p^- = 0$ it follows that,

$$P\{(M,0)|(M,0)\} = (1 - p_{ab} - p_{af})^{M-M} = 1 \quad (4)$$

.. and $(0 \leq P\{(v,b)|(v,b)\} < 1)$.

I.e. the only diagonal element equal to unity is $(M, 0)$. This means that our model is an absorbing Markov chain and $(M, 0)$ is the only absorbing state. Stochastic search will therefore eventually reach the state $(M, 0)$ in finite time and then will stay in this state forever. All other states are transient. The rate of convergence is geometric for some constant c, $(0 < c < 1)$.

The above proposition proves the intuition that in the presence of the target in the search space all agents will eventually converge on its position. Thus we see that indeed in this case the parameters a and b do not influence the convergence of SDS.
In the situation when the target is not present in the search space the following result can be proven.

Proposition 2. *Given $p^- \neq 0$ the strong convergence criterion does not hold in the stochastic diffusion search.*

Proof. We will prove the above assertion by showing that a less restrictive property, of which strong convergence criterion is a special subclass, is not fulfilled either. We will show by contradiction, that

$$\exists_{a,b>0}(2b < M \wedge b + a \leq M \wedge a - b \geq 0) \tag{5}$$

$$P\{\lim_{n\to+\infty} P\{|z_n - a| < b\} = 1 \tag{6}$$

.. is not true.

Suppose the above assertion holds. It is equivalent to:

$$\exists_{a,b>0}(2b < M \wedge b + a \leq M \wedge a - b \geq 0) \tag{7}$$

$$P\{\lim_{n\to+\infty} P\{|z_n - a| \geq b\} = 0 \tag{8}$$

Let $p^- \neq 0$. In the case of $p^+ = p_d = 0$ the ephemeral states with a nonzero amount of noise are excluded from consideration. From the probability transition matrix it follows that for any state $(i, j) \in S$,

$$P\{S_{n+1} = (i, j)|S_n = (0, 0)\} > 0 \tag{9}$$

and

$$P\{S_{n+1} = (0, 0)|S_n = (i, j)\} > 0, \tag{10}$$

i.e. the first row and first column of the transition probability matrix $\mathbf{P}$ are strictly positive.

It follows that any entry of $\mathbf{P}^2$ is positive, hence $\mathbf{P}$ is primitive. From the Perron-Frobenius theorem it follows, that there exists over states in S a limit probability distribution p_∞, such that $p_\infty > 0$. This implies that in a steady state all of the states occur with probability strictly greater than one, i.e. infinitely often and $z_n = max_{s\in S}(f_n^s)$ takes all possible values from the set $\{0, ..M\}$ with positive probability. This contradicts our assumption.

From the above proof it follows, that in the case of $p^- \neq 0$ the model of stochastic diffusion search is an ergodic Markov chain [58]. Therefore it is easy to see that stochastic diffusion fulfils another, weaker convergence property stated in proposition 3.

Proposition 3. *Given $p^- \neq 0$, stochastic diffusion search converges in a weak sense, i.e.*

$$(\exists a > 0)(\{\lim_{n\to\infty} EZ_n = a\}). \tag{11}$$

Proof. Follows immediately from the ergodicity property of stochastic search and observation that z_n is a random variable defined on the probability space $(S, \sigma(S), p_n)$, where $\sigma(S)$ is a σ-algebra of all subsets of S.

The above characterisations show that in the most general case Stochastic Diffusion convergence has to be understood as approaching an equilibrium in a statistical sense. It means that even after reaching a steady state all possible configurations of agents pointing to the best instantiation of the target as well as to disturbances occur infinitely often according to limiting probability distribution (however some of them may occur very rarely). In practice with appropriate estimates for halting parameters a and b the algorithm will stabilise for a long enough period thus enabling termination.

Also the convergence in the weak sense is crucial for the adaptability of SDS. Effectively SDS allocates a certain amount of computational resources to the best solution found so far. Remaining resources explore the search space attempting to discover other potential solutions. A given optimal solution can become suboptimal in two ways - either its fit to the target decreases because the similarity criteria change over time or a new, better solution appears in the search space over the course of time. In both cases SDS will be able to find a new optimal solution (due to the agents exploring the search space) and once found, it will rapidly reallocate most of resources towards the new optimal solution.